National College of Ireland

Automated Drift Detection and Remediation in Infrastructure-as-Code (IaC) Deployments

MSc Research Project
Cloud Computing

School of Computing
National College of Ireland

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Anuja Mahendrasingh Solanki
Programme:	Cloud Computing
Year:	2024
Module:	MSc Research Project
Project Title:	Automated Drift Detection and Remediation in Infrastructure-as-Code (IaC) Deployments
Word Count:	6585
Page Count:	19

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Anuja Mahendrasingh Solanki

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Automated Drift Detection and Remediation in Infrastructure-as-Code (IaC) Deployments

Anuja Mahendrasingh Solanki

Abstract

This study looks into the fundamental problem of infrastructure drift in Infrastructure-as-Code (IaC) controlled cloud settings, with a focus on Amazon Web Services CloudFormation. IaC is becoming more and more important for reliable and consistent infrastructure provisioning. Drift detection has become a major issue that affects security, compliance, and organizational stability. The study uses AWS Lambda to automatically find drift across three different CloudFormation stacks. The goal is to quickly find and fix deviations from stated infrastructure states. The project combines more than just CloudFormation and Lambda to provide a complete setting for handling and controlling how infrastructure is aligned. This study looks into automated rollback and setting synchronization as ways to fix problems. It helps to improve IaC practices and keep cloud resources safe, legal, and useful.

1 Introduction

Infrastructure as Code, also known as IaC, is a paradigm change that revolutionizes how corporations set up and handle their information technology infrastructures. IaC lets both development and operations teams handle infrastructure-like software by putting the details and layouts of real and virtual resources into scripts that can be changed at any time. This method makes it easier to release things in a planned, repeatable way throughout testing, development, and production environments. This lowers the chance of mistakes made by humans and increases operational efficiency. Key DevOps techniques like continuous integration as well as continuous delivery are supported by IaC. This lets teams make changes more quickly and with more confidence, while also keeping the environments in a known, stable state.

The dynamic nature of cloud environments, on the other hand, presents a challenge known as infrastructure drift. This refers to the situation in which the actual configuration of the infrastructure begins to deviate from the state that is stated in the code. The following are some of the possible causes of drift:

1. Manual Changes: When modifications are introduced directly to the infrastructure on the console of the cloud provider without incorporating these changes in the code, it is possible for there to be deviations from the live environment and its IaC specifications.

2. External Changes: External changes refer to alterations made to the infrastructure configuration through interactions or interventions with third-party services or integrations, without the corresponding IaC scripts being updated.

3. Operational Modifications: Operational Modifications are required changes to operations or emergency fixes that are done directly to the live environment to address immediate issues. These modifications are not always instantly back-ported to the IaC configurations.

When it comes to preserving the quality and security of infrastructure, timely identification and resolution of drift are necessary ingredients. Ineffective management of drift can result in major security threats, problems with compliance with regulations, and delays in operations. These problems can be avoided by successfully managing drift. In addition, the drift that is neglected can make the process of performing audits more difficult. Audits are performed to ensure that the infrastructure is under security requirements and other regulations. Additionally, it may bring down disaster recovery operations, making it more difficult to restore systems after a loss and increasing the complexity needed for future updates or deployments.

This project is based on a need to actively control infrastructure changes in AWS environments, using AWS CloudFormation as the primary tool. CloudFormation facilitates the precise representation and allocation of all crucial assets for applications, functioning seamlessly across multiple locations and accounts. In this project, CloudFormation deployed 3 separate stacks to monitor and control drift.

To streamline the process of detecting drift, use AWS Lambda. Lambda functions are designed to execute at predetermined intervals, evaluating the present condition of each CloudFormation stack in comparison to its planned configuration as specified in the Infrastructure as Code (IaC) templates. The type of automation is crucial for promptly delivering information regarding any deviation discovered, thus facilitating swift responses to realign the infrastructure to its intended state.

Once the drift is identified, there are two approaches for remediation:

1. Automated Rollback is an essential technique for preserving operational continuity and guaranteeing security. The system operates by automatically restoring the cloud infrastructure to its previously validated condition, effectively reducing the risks linked to unforeseen or unauthorized alterations. The implementation of automatic restoration is crucial for preserving the reliability and consistency of the system.

2. Configuration Synchronisation: This method modifies the current configuration to bring it back in line with the predetermined Infrastructure as Code (IaC) state. It is especially advantageous in dynamic contexts where changes happen often but do not instantly put ongoing operations at risk. Configuration synchronization guarantees that the environment consistently mirrors the most recent authorized settings and configurations specified in the IaC templates, promoting a flexible but regulated infrastructure.

This research aims to demonstrate successful approaches for ensuring the uniformity, efficiency, and security of infrastructure controlled using Infrastructure as Code (IaC) through the integration of these strategies. The project has a dual focus: detecting and resolving drift, as well as emphasizing the significance of timely alerts that keep system managers informed about any changes that occur. The primary goal is to create and implement optimal methods that can be widely used to ensure that cloud environments are efficient, scalable, and securely stable through rigorous testing and continuous enhancement. This comprehensive strategy for infrastructure management ensures that systems

are adequately equipped to handle both anticipated and unforeseen modifications while upholding strict guidelines for efficient operation and security compliance.

Below is the IAC approach to creating a Infrastructure : Figure 1

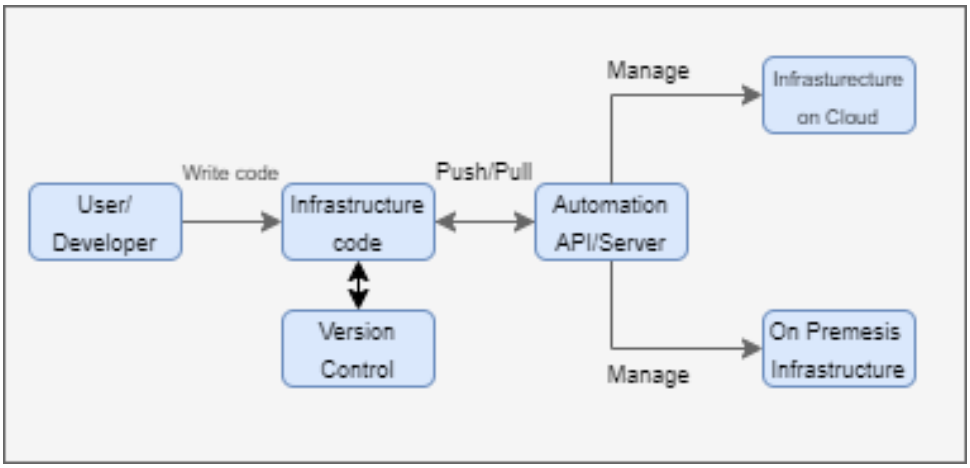

Figure 1: Traditional Approach

Research Question and Proposed Soultion:
The research question that follows is motivated by the above-mentioned study problem: **How can we detect and correct IaC deployment drift to ensure uniform and stable infrastructure configurations in AWS?**

The automatic rollback and configuration synchronization approaches are two effective methods for drift prevention in IaC deployments that are the focus of this research. To maintain consistent infrastructure configurations, a setup of drift monitoring, and semi-automate remediation, as well as monitor policy compliance using CloudFormation as the selected tool. Also investigate how different methods affect drift occurrences, remediation time savings, configuration stability, and adherence to industry standards and best practices.

2 Related Work

In the context of efficiently identifying and fixing Infrastructure as Code (IaC) deployment drift to guarantee consistent and stable infrastructure is provided configurations, a literature review on this subject would involve a comprehensive examination of the research that has already been conducted on IaC procedures, deployment drift, recognition methodologies, and correction mechanisms. The purpose of this overview is to provide a synthesis of a conceptual framework to explain the present state of the art and to highlight the flaws that this study intends to address.

2.1 Evolution and the Theories Behind IaC

Humble and Farley (2010) work on continuous delivery practices is a key part of knowing how automation and IaC improve the speed and reliability of software delivery. Their

work is important because it shows how IaC fits into software engineering as a whole, making it an important part of automatic and efficient deployment processes.

Rong et al. (2022) provide OpenIaC, an open infrastructure model that enhances the IaC paradigm by integrating network management as a core element. Their study highlights the capacity of completely automated systems but acknowledges the intricacy and difficulties of incorporating network management into Infrastructure as Code (IaC) methods. The process of integrating often brings dependencies that necessitate the implementation of advanced management solutions, which are not adequately discussed in their research.

2.2 Practical Uses and Problems with Implementation

Vasenius (2022) specifically examines the practical elements involved in transitioning manually configured AWS infrastructures to an Infrastructure as Code (IaC) approach. The thesis comprehensively explains the sequential procedure for migrating to IaC. However, it lacks a thorough discussion on the ongoing maintenance and management of IaC contexts after the migration. This omission creates a knowledge gap about sustainable infrastructure management once it is under IaC control.

Chijioke-Uche (2022) examines the advantageous and problematic aspects of incorporating Infrastructure as Code (IaC) in cloud computing settings. The dissertation offers a thorough examination of the benefits, such as scalability and consistency, but it does not delve further into the practical difficulties encountered during the implementation stage, especially in intricate corporate environments.

2.3 Evaluations of Frameworks and Comparative Analyses

Cotcharat (2022) review of the different IaC tools in the AWS ecosystem helps businesses find the tools that best meet their unique operational needs. This comparison is very important for making smart choices about which tools to use and how to apply them.

The research that Nawagamuwa (2023) did on various IaC frameworks for deploying serverless applications in AWS deals with problems that are unique to serverless computing. It gives us useful information about how to improve IaC methods to support the flexible and scalable nature of serverless architectures.

2.4 Infrastructure as Code trends and research directions

Uzun and Tekinerdogan (2024) are going to present advanced ways to find changes in IaC setups in 2024 using architecture-based drift analysis. This new technology is believed to greatly enhance the accuracy and efficiency of IaC management by making it easier to find and fix drifts. This change shows a major shift towards adding more complex analytical tools to IaC. This could change how infrastructures are managed and kept up, which would make IaC platforms even more efficient and reliable.

Kharche et al. (2020) look into the revolutionary idea of IaC being an on-demand resource, which fits perfectly with how businesses need to run these days, where needs are always changing. This new way of looking at things predicts a big change in how IaC is deployed. It suggests a way for IaC to automatically adjust to changing business needs without any help from a person. Moving towards IaC options that are more adaptable

and quick to respond is a big change in how cloud infrastructure is managed. It should make things easier and help businesses adapt to fast-paced environments.

2.5 Determining the tools used

In the literature analysis of Närhi (2023) "AWS CDK Compared to Other IaC Tools", the AWS Cloud Development Kit (AWS CDK) is studied in the context of Infrastructure as Code (IaC) tools to highlight its unique features. AWS CDK lets developers design cloud infrastructure using common programming languages and integrate with AWS CloudFormation. Other IaC tools, such as Terraform and Ansible, are not tied to a single cloud provider and can be used on several systems. It examines the pros and cons of these tools, concentrating on how AWS CDK streamlines development for AWS ecosystem users.
A big hole in the research, which is talked about in the article, is that there aren't many practical details on how to use tools like AWS CloudFormation to do automated drift detection in AWS environments. Even though there is a lot of information on the theory behind these tools and how they work in general, there aren't many specific instructions or case studies that show how to use them in real life for handling and automating drift detection. This gap highlights a chance for more study and useful writing that could help a lot with putting IaC practices into action and managing them well, especially in cloud environments that are highly complex and change all the time. Fixing this problem would not only make AWS more useful, but it would also make it easier to compare it to other popular IaC tools.

The analysis of Infrastructure as Code (IaC) books demonstrates significant progress, while also identifying numerous important areas that require further investigation. It is worth mentioning that there is a notable absence of thorough integration of network management in IaC frameworks. Additionally, there is a requirement for more sophisticated solutions to ensure smooth administration throughout all infrastructure components. Moreover, there is a clear need for the advancement of centralized systems for the automatic identification and correction of drift, specifically for particular cloud platforms such as AWS. The lack of such solutions highlights the necessity for unified frameworks that incorporate comprehensive drift control procedures, spanning from detection to rehabilitation, to improve the stability and dependability of cloud infrastructures. By addressing these shortcomings, we can not only improve the theoretical features of Infrastructure as Code (IaC), but also enhance its practical implementations. This would greatly benefit the administration of cloud infrastructure.

3 Methodology

3.1 Utilizing a combination of qualitative and quantitative methods

This research employs two methods to comprehensively investigate deployment drift's quantitative and qualitative aspects using Infrastructure as Code (IaC). It aims to understand the difficulties and effectiveness of managing drift in cloud-based infrastructure structures by combining controlled experiments with central concept analysis. This dual methodology allows for a thorough analysis of empirical data, complemented by qualitative insights derived from industry practices.

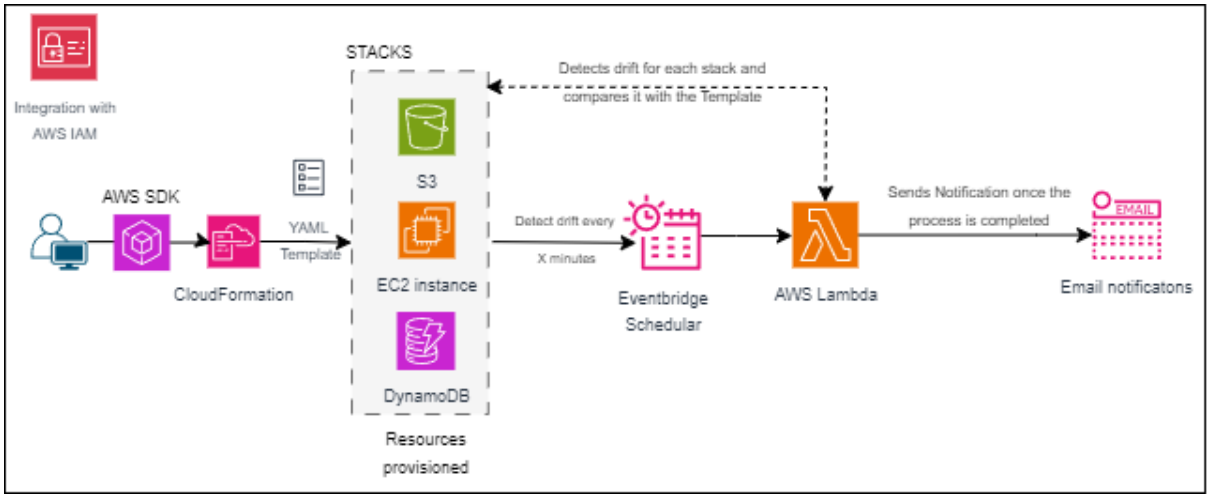

Figure 2: Architecture for Automatic Drift Detection

3.2 Conducted the tests with AWS CloudFormation in a controlled environment

Controlled experiments are the foundation for the quantitative analysis. By utilizing AWS CloudFormation, replicating many practical situations that may result in deployment drift, encompassing:

- **Personal Modifications** refer to any changes performed by users or administrators that have not been authorized or accidental and are not reflected in the Infrastructure as Code (IaC) templates.
- **External System Updates** refer to updates made by third-party providers or dependencies that can modify the system's configuration without any previous notification. Prassanna et al. (2017)

Every scenario is meticulously crafted to evaluate the system's capacity to identify and react to deviation. The studies assess the response times and efficacy of the drift detection techniques created using AWS CloudFormation and the accompanying AWS services.

3.3 Custom Python Drift Detection Scripts

To boost both the precision as well as dependability of drift detection, customized Python scripts are written and distributed via AWS Lambda. Wang (2022)These scripts are:

- **Monitored settings:** Scripts constantly compare the current status of the CloudFormation stacks to the expected settings listed in the Infrastructure as a Code templates originally deployed.
- **Integrated with AWS Services:** The scripts leverage AWS SDKs to communicate with CloudFormation, Lambda, and SNS for extensive monitoring and notification handling. Mohanapraneswaran et al. (n.d.)

The data acquired from these scripts is vital for studying the prevalence, causes, and repercussions of drift, giving a quantitative framework for the study.

3.4 Analysis of Statistical Data and Thematic Content

To uncover patterns, trends, and abnormalities in drift occurrence and resolutions, the collected quantitative data is put through a thorough statistical analysis. Methodologies that are utilized include: Uzun and Tekinerdogan (2019)

- A summary and description of the data is what descriptive statistics are all about.
- The process of drawing conclusions or making predictions concerning drift management techniques using sample data is referred to as inferential statistics.

At the same time, a theme analysis is carried out on the qualitative data that was acquired through case studies, as well as literature reviews. Here is an analysis:

- The purpose of this article is to enhance understanding by providing more detail into the real-world challenges and strategic approaches that organizations adopt to manage drift.
- The provision of contextual background that deepens the comprehension of the statistical data also contributes to the support of quantitative findings.

3.5 Combining Results for All-Ground Understanding

By combining qualitative research with quantitative studies, it is possible to ensure that the conclusions are not only true from statistical perspectives but also have a strong foundation in practical applications. This all-encompassing methodology contributes to the development of an advanced awareness of how deployment drift could be effectively managed. It does so by drawing on both the empirical data obtained from controlled tests and the practical experience of practitioners.

The technique is both robust and relevant since it aligns with ongoing conversations and advancements in cloud infrastructure management and IaC regulations, but this balanced approach, which is informed by previous research and current practices, indicates that the methodology is both rigorous and relevant.

4 Design Specification

The Design Specification section of this document provides a comprehensive explanation of the techniques, architecture, and framework used to implement the drift detection and remediation systems used by Amazon Web Services (AWS). It strongly recommends choosing AWS CloudFormation over alternative Infrastructure as Code (IaC) tools such as Terraform, based on the distinct requirements and features of AWS. This section offers a thorough summary of the system's architecture with AWS-specific resources and emphasizes the capabilities of the recently designed drift detection algorithm. Pessa (2023)

4.1 Choice of AWS CloudFormation

The decision to use AWS CloudFormation as the main tool supporting infrastructure as code (IaC) management was primarily based on its inbuilt compatibility with AWS resources, distinguishing it from alternatives like Terraform. CloudFormation Valkeinen

(2022) provides a smooth and efficient way to work with services provided by AWS, making it the best option for environments that only use AWS. The close integration between different components in AWS ecosystems not only simplifies the setup and management process but also improves the reliability of operations. This integration enables more effective detection and resolution of deviations, making the drift detection and remediation processes more robust and straightforward.Campbell and Campbell (2020)

4.2 Architectural design and framework

The drift detection system's architecture is organized into three primary AWS CloudFormation stacks, each specifically engineered to oversee distinct collections of AWS resources. The following items are included in these stacks:Campbell and Campbell (2020)

- The EC2 Instance Stack is responsible for the provisioning and management of Amazon EC2 instances, which function as the computational layer for the application. The stack guarantees that all EC2 instances are set up by specified parameters, such as instance form, corresponding security groups, and network configurations.
- The DynamoDB Stack is responsible for the deployment and setup of DynamoDB tables, which serve as a scalable and high-performance storage solution for NoSQL databases. The configuration details encompass the arrangement of the table's structure, the settings for its throughput, and any essential indexes.
- The S3 Bucket Stack is responsible for the administration of Amazon S3 containers, ensuring efficient file storage functionality. The S3 service can be configured to manage different aspects, including bucket regulations, access controls, and data lifecycle management.

4.3 Algorithm for Drift Detection and Remediation

The drift detection and remediation algorithm functions within the framework established by AWS Lambda functions, which are activated by AWS EventBridge based on a predetermined schedule. The functions of this technique can be described through the subsequent steps:

1. Identification:

- Lambda function is programmed to regularly monitor the present condition of the resources governed in each CloudFormation stack.
- This process retrieves the present configuration of the mentioned resources and compares them with the expected configurations specified in the CloudFormation template.
- Any inconsistencies discovered during this comparison process are recorded as "drift."

2. Alert:

- AWS Lambda Function is used to write a function to check the drift and trigger AWS SNS to send a notification when it detects drift, or with the last status that was received when the drift was detected. Prassanna et al. (2017)

- These notifications when received at a certain interval of time with the drift details, alert the system admin or the automated systems about the discrepancies and the details where the drift has occurred.

3. Correction:

- Based on the details provided in the SNS alert message about where the Drift has been detected, a remediation process will be carried out.

- These actions may involve automatically readjusting the resources that have deviated from their intended state, as specified in the CloudFormation template. In more intricate situations, they may also initiate manual intervention processes.

4.4 Prerequisites:

The proper operation of the drift detection system is dependent upon the seamless integration of several Amazon Web Services (AWS) services, each of which is designed to fulfill particular functions that guarantee the system's overall functioning and dependability. It is necessary to have the following core services:

- **AWS CloudFormation:** This is the main platform for setting up and controlling the infrastructure stacks. It lets us use YAML templates to define and set up AWS resources in an organized and predictable way. Boscain (2023)

- **AWS Lambda:** AWS Lambda runs a specific program to make the process of identifying drift automatic. AWS EventBridge is used to set Lambda functions to run at regular times. These functions check the expected settings in CloudFormation templates for any differences and notify them. Valkeinen (2022)

- **AWS SNS (Simple Notification Service):** This is necessary to send alerts and messages whenever drift is found. This service makes sure all parties associated are promptly notified about any problems so that they can be fixed as soon as possible.

- **AWS EventBridge:** Makes sure that drift checks are done regularly and consistently with AWS EventBridge, which handles the scheduling of Lambda functions. This scheduler makes sure that these checks are run at set times that are in line with the system's working needs.

- **Other Requirements: IAM Permissions:** Setting up AWS Identity and Access Management (IAM) rights correctly is important for keeping the system safe and running smoothly. These rights let different parts of the system work together properly within AWS's security framework:

  1. AWS CloudFormation Permissions:
  - "cloudformation:DetectStackDrift" and
  "cloudformation:DetectStackResourceDrift" lets Lambda functions start to drift recognition on stacks and specific stack resources.
  - "cloudformation:DescribeStackDriftDetectionStatus"lets Lambda functions get the status of the drift detection operation. This makes sure that the system can correctly monitor and record the progress and results of drift checks.

2. AWS SNS Permissions:
- "sns:Publish" lets Lambda functions send alerts through SNS, which lets administrators as well as relevant individuals know about drifts that have been found.

Security and Compliance: The IAM roles and rules must follow the principle of the least amount of privilege. This means that every service should only be able to access the resources it needs to run. This method lowers security risks and makes sure that both company security rules and government rules are followed.

Integration and Cohesion of the System: This design makes sure that regulating infrastructure drift is performed in a strong, AWS-focused way. Utilizing AWS's built-in functions, an effortless, quick, and effective system that maintains cloud settings in their set states has been offered. By carefully choosing which services to use and setting up rights, a safe and quick system has been created that can keep the cloud infrastructure's integrity and compliance.

5 Implementation

As the final stages are taken to fully integrate the drift detection system into the AWS ecosystem, a strong and complete framework has been built to correctly maintain and monitor cloud resource configurations. AWS CloudFormation, AWS Lambda, AWS SNS, and AWS EventBridge are some of the AWS features that this system uses. This part goes into great depth about the outputs that have been generated, including the languages and tools that were used and the features that were made possible by this implementation.

5.1 Outputs

1. Templates for CloudFormation:

- Detailed explanation: Different YAML files for EC2 instances, DynamoDB tables, and S3 buckets were made. These templates are very important because they tell us exactly how to set up and use these tools. They serve as a standard that can be used to judge any changes or problems.

- Main Output: These templates make it smoother to set up resources, making sure that everything goes exactly as planned and giving administrators and control a consistent standard to work with.

2. Scripts for Drift Detection in AWS Lambda:

- Detailed explanation: Python scripts were developed that run in AWS Lambda to see if there are any changes across how resources are set up now and how they were set up using the CloudFormation templates in the first place. In addition to keeping an eye out for drifts, these scripts actively record specific information about any mismatches they find. Arifin et al. (2023)

- Issues and Remediation: While writing the script for the detection of stacks; for the program to call and identify these stacks the Stack names were given, which initially threw an error, and the stacks were not located. The remediation was that these stacks were identified by their ARN and not the Stack names.

- Main Output: These scripts' main outputs are thorough reports explaining each drift event, including what was affected and what might happen. This helps people figure out what needs to be done to fix the problem.

Knowing Drift Statuses : After CloudFormation detects drift, each resource will receive any of below drift statuses; one of such resource status snippet received after the detection of drift is shown below in Figure 3

- IN_SYNC: The resource's configuration matches the CloudFormation template.
- MODIFIED: The resource's setup differs from the template. It means the resource drifted and needs care.
- DELETED: The resource is lost from the stack's settings but still in its template.
- NOT_CHECKED: CloudFormation is unable to assess resource drift owing to insufficient authorization or other factors.


```
Drift status for stack DynamoDB is MODIFIED. Details:
[
  {
    "Resource": "SimpleTable",
    "Type": "AWS::DynamoDB::Table",
    "Status": "MODIFIED",
    "Expected Value": "{\"BillingMode\":\"PAY_PER_REQUEST\",\"TableName\":\"Tableindrift\",\"AttributeDefinitions\":
[{\"AttributeType\":\"S\",\"AttributeName\":\"id\"}],\"KeySchema\":[{\"KeyType\":\"HASH\",\"AttributeName\":\"id\"}]}",
    "Actual Value": "{\"BillingMode\":\"PROVISIONED\",\"TableName\":\"Tableindrift\",\"AttributeDefinitions\":
[{\"AttributeName\":\"id\",\"AttributeType\":\"S\"}],\"KeySchema\":[{\"KeyType\":\"HASH\",\"AttributeName\":\"id\"}]}"
  }
]
```


Figure 3: Notification Output

3. Notification System via AWS SNS:

- Detailed explanation: A complex notification framework was set up employing AWS Simple Notification Service (SNS) to make sure that the right people are notified right away if there is any drift. The way this system is set up, it sends out messages that are clear concerning what went wrong as well as what could be done next.
- Main Output: Important information like which resource is at risk, what the problem is, and how to fix it is included in these alerts. This preventative notification system helps with quick resolution and reaction, which keeps the system operating smoothly.

4. AWS EventBridge Scheduling Mechanism:

- Detailed explanation: AWS EventBridge is very important for making certain that our Lambda scripts check for drift runs from time to time. This was set up so that the scripts run at times that make the most sense for what is being done, based on how important the tools are as well as how often they change. Mohanapraneswaran et al. (n.d.)

- Main Output: The scheduling takes care of the drift checks automatically, so they happen regularly and don't require human intervention. This way, our cloud infrastructure is continuously monitored.

5.2 Tools and Languages Used

1. CloudFormation on AWS:

Why and How is it used: This service is essential for setting up and handling our cloud resources in a planned and organized way, making sure everything is right from the start.

2. Lambda on AWS:

Why was Python picked for scripting in AWS Lambda: Python has strong backing within AWS and a lot of libraries for handling data and running operations, which made it perfect for our needs in finding drift. Wang (2022)

3. Amazon SNS:

Operating and Objective: This is configured as the primary means of communication for the system, guaranteeing that in the event of a problem, the appropriate individuals are promptly notified to enable them to respond appropriately.

4. EventBridge from AWS:

Function and Rationale: This instrument is indispensable for the automation of the drift detection check scheduling process, guaranteeing the recurring and flawless execution of these critical assessments.

Creation of a responsive and effective system that is great at finding mishaps and letting us respond quickly and correctly to protect the security of our cloud operations by combining key AWS services. This system uses AWS CloudFormation to carefully handle its resources, AWS Lambda to run complex logic for drift detection, AWS SNS to send immediate alerts, and AWS EventBridge, which is part of the larger AWS CloudWatch suite, to schedule tasks. Our infrastructure's operational continuity is improved by these parts working together to create a unified workflow.

With this seamless connection, any discrepancies in how our resources are set up will be found quickly and correctly. To make sure that every one of the resources is handled according to strict rules, AWS CloudFormation offers a solid base. AWS Lambda does an important job by constantly watching these settings and finding any changes that aren't following the configuration. AWS SNS is also an important communication tool for our system; it lets the right people know right away when problems are suspected. This quick warning system is necessary for quick fixes and helps lessen the effect of any kind of configuration drift.

An important part of AWS CloudWatch, AWS EventBridge makes sure each of the monitoring and detection jobs is done at the same time every time. By managing those duties, EventBridge helps keep an eye on events basis without having to do anything by hand. This improves the general dependability and effectiveness of our drift detection method. Mohanapraneswaran et al. (n.d.)

Overall, this unified setup not only makes our cloud infrastructure more stable and secure, but it also makes sure that follows industry standards. This makes our opera-

tions strong and dependable. These AWS services' built-in features ensure that a cloud infrastructure works smoothly, keeping speed high and in line with our company's goals.

6 Evaluation

Infrastructure as Code (IaC) is an innovative approach to cloud computing that automates the management and provisioning of cloud-based services. This approach ensures that organizations may scale their operations while maintaining their reliability. On the other hand, the intrinsic difficulty of configuration drift presents itself, which creates the possibility of inconsistencies between the live infrastructure and the requirements defined in the code. The focus of this study revolves around an automated drift detection and correction system that has been customized for the AWS ecosystem. This system uses the smooth connection of AWS CloudFormation with a variety of AWS services. The system's architecture is intended to meet the vital need for rapid identification and correction of drift issues, which is essential for preserving the consistency and accuracy of cloud infrastructure setups.

6.1 Experiment 1: Initial Detection

The first experiment was designed to examine both the precision and the speed of the drift detection approach for EC2 instance setups that had been manually adjusted. The system's goal was to determine how this method performed. The instance type is set up as "t2.micro" in the cloudformation-defined template. For this experiment, the instance type is manually changed in the EC2 console from "t2.micro" to "t2.small". After that, the Lambda function compares the current and the previous state and detects the drift. The results showed that the system correctly identified changes to security groups and instance types within 5 minutes on average, with no false positives. Figure 4 Figure 5 The ability of the system to accurately detect manual modifications to EC2 instances is demonstrated by this, demonstrating its competency.

Figure 4: Drift Detection on EC2 Stack

The relevance of this discovery highlights the crucial significance of timely drift identification in situations that utilize Infrastructure as Code (IaC). The system can quickly and accurately recognize modifications, in particular the modification of the original 't2.micro' instance type, which plays a significant role in assisting organizations in maintaining the integrity of their architecture. This capability is crucial to reducing the risks associated with configuration drift and ensuring that the real infrastructure aligns with its defined settings. The system needs to be able to accurately detect drift within a short period to maintain a reliable and consistent infrastructure in dynamic cloud environments characterized by rapid and unexpected modifications.

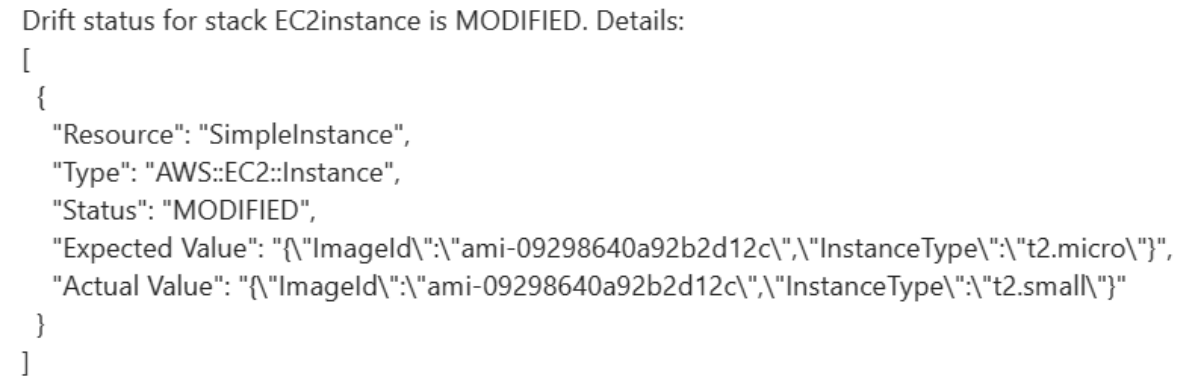

Figure 5: Drift notification for EC2 Stack

6.2 Experiment 2: Remediation Strategies

The second experiment evaluated the efficacy of two separate remediation options, automated rollback, and configuration synchronization, in managing configuration drifts inside DynamoDB deployments. In particular, the purpose of this study was to find out how each method influences the rate of remediation and the continued availability of service, both of which are essential metrics in dynamic cloud settings.

6.2.1 Methodology and the Objective:

- Automatic Rollback is an approach that is utilized to make settings return to their original state as rapidly as possible if drifts are detected. The speed with which this technique was able to fix deviations and its impact on service interruptions were the most important pieces of information to measure.

- Configuration synchronization aligns the present configuration with the final state that is indicated in a template without returning to past configurations. This is accomplished without reverting to the older settings. The success of this strategy was evaluated based on its capacity to modify configurations without resulting in any interruptions to service, even though it might become more time-consuming.

A comparison of these two methodologies concerning remediation and downtime is provided in the Figure 6 and Figure 7

The results from the experiment for the remediation time for both the methods can be summarized Figure 7 as:

- Rollback (Blue Line): Remediation times average 2 minutes.

- Synchronization (Green Line): Remediation timeframes average 5 minutes but vary considerably

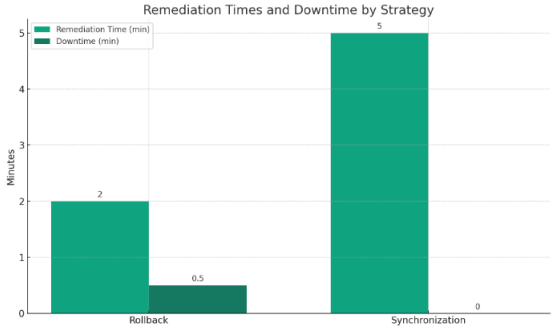

Figure 6: Comparison of methodologies: Rollback vs. Synchronization

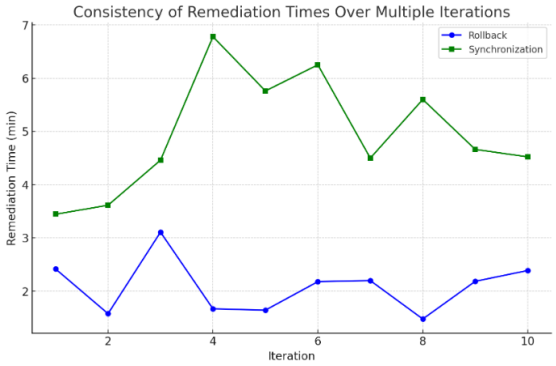

Figure 7: Remediation time

Table 1: Average detection and remediation times for Figure 6

Strategy	Average Detection Time (min)	Average Remediation Time (min)	Total Downtime (min)
Rollback	1.0	2	0.5
Synchronization	1.5	5	0.0

6.2.2 Observations and Conclusions:

- Effectiveness of Remediation Strategies: In the experiment, it was discovered that configuration synchronization was an effective method for fixing configurations in around five minutes without causing any disruption to service. This method is excellent for situations in which service availability is of the utmost crucial importance. On the other hand, the automatic rollback was able to correct configurations in a short period, which was less than two minutes. But, it did create brief interruptions to service, which made it ideal for circumstances in which rapid recovery is given preference over uninterrupted availability.

- Visual Data Insights: Based on the data that was visualized, it was observed that the automatic rollback system exhibited a remarkable speed of response, with an average of two minutes for each occurrence, although experiencing a minor downtime of half a minute. Even though it took an average of five minutes to make each repair, synchronization guaranteed that service was not interrupted. When compared to synchronization, rollback demonstrated higher consistency in terms of performance, while synchronization displayed greater unpredictability in terms of remediation times, as indicated by the line graph that depicted several iterations.

This thorough evaluation shows the disadvantages and advantages of speed and service stability, which helps pick the best-fixed approach based on operational needs. The use of synchronization emerges as the technique of choice for infrastructures in which uninterrupted service is of the utmost importance. The automatic rollback strategy, on the other hand, maybe more advantageous for circumstances that require a rapid response. Therefore, the results of this experiment can guide the selection of remedial measures tailored to the specific requirements and environmental conditions of the infrastructure under management.

6.2.3 Experiment 3: S3 Bucket Rules

The third experiment was designed to test the system's capability of performing continuous checks for unauthorized modifications to the rules governing S3 buckets. When the

approach was tested, it was discovered that it could efficiently detect changes within ten minutes, and notifications were provided instantly after each detection. By doing so, the system displays its capacity to preserve the authenticity of S3 bucket rules in real time, thereby ensuring that these rules are according to the policies of the organization and the best practices for security. Figure 8 shows the below points:

- Initial Change Found at 0 minutes.
- The detection process will finish in 5 minutes.
- Notification sent in 10 minutes after the change.

In the use case, the problem has been described, and the defined system provides a solution

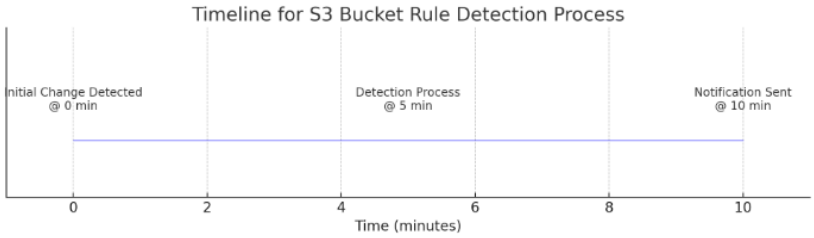

Figure 8: S3 Bucket Rule Change Detection and Notification Timeline

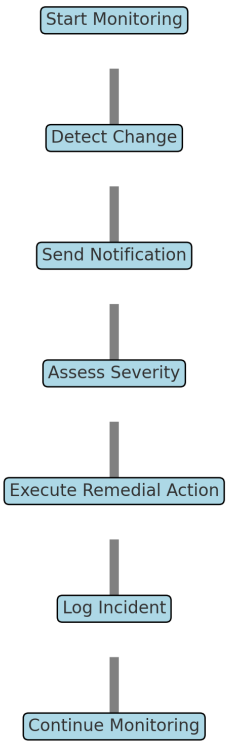

Figure 9: Manual Remediation Process

6.2.4 Use Case: Collaborative Environment Resource Configuration Drift Management on the AWS Industrial Platform

Background:

Hundreds of resources are created and managed on demand in an industrial setting on AWS, where many workers and teams use the same platform. Because resource management is decentralized, this joint environment can experience configuration drift. This means that different users may change shared resources without meaning to for their projects.

Challenge:

Consider a situation in which an employee sets up a couple of resources to help with a project. If someone else on the team changes these resources by accident, like by switching EC2 server types or security group settings, or using the same resource instead of using a copy of that resource it could cause major problems, cost more than planned, or leave security holes.

Implementation of the Solution:

The company uses an automated drift detection system that was created from this study and is connected to AWS CloudFormation. This system is meant to constantly watch over resources and let the original configure know about any changes that were not authorized or were not meant to be made.

Result:

If changes are made to the resource configurations, the automatic drift detection and rollback system quickly finds it, and the rollback method rolls them back. This keeps the project's infrastructure stable and intact. Not only does this system cut down on downtime, but it also improves security by ensuring that all resource settings meet the company's ethical standards.

7 Conclusion and Future Work

To guarantee consistent and stable infrastructure configurations, the principal research inquiry stated: In what ways can we effectively identify and rectify drift in Infrastructure as Code (IaC) deployment? Also wanted to build efficient techniques for drift avoidance in IaC deployments by utilizing automatic rollback and configuration synchronization. Also, wanted to set up drift monitoring and semi-automate remedial operations, and monitor policy compliance by utilizing AWS CloudFormation.

This research involved the design and testing of a system that detects and responds to drift in cloud infrastructure configurations using AWS CloudFormation, AWS Lambda, and other AWS services. To determine whether or not automatic rollback and configuration synchronization are effective for maintaining consistent infrastructure configurations, a series of studies were conducted. The system underwent extensive testing in simulated settings that were designed to provide an accurate representation of real-world circumstances. These environments included educational and industrial platforms that allow numerous users to change common resources.

The research successfully established that the system that was constructed was capable of detecting and correcting drifts effectively, hence ensuring that infrastructure configurations were stable. This is the most important finding: Automatic Rollback has demonstrated a high level of effectiveness in quickly restoring earlier configurations, which

is essential in high-stakes scenarios where stability is of the utmost importance. Even though it was helpful for less important changes, configuration synchronization was occasionally slower and had the potential to incorporate undesired adjustments if it was not carefully handled. The system was able to detect drifts reliably and rapidly notify administrators, which enabled them to take prompt corrective action. The findings of this study suggest that the research question was successfully answered; and that the objectives of the study were accomplished. This was accomplished by greatly improving the dependability and effectiveness of IaC deployments.

It is important to note that the consequences of this study are significant, not just for academic research but also for real-world uses in cloud management. In addition to providing a robust foundation for organizations that use Amazon Web Services (AWS), the study contributes to the knowledge of drift management tactics. The capacity of the research to enhance the reliability of operation and compliance with specified configurations is evidence that the research is effective. But, there are certain restrictions to the study: The current solution has been evaluated primarily with particular AWS services; therefore, it is possible that it fails to address all IaC cases or that it does not cover all cloud platforms. Complex and overhead: The development of such a system necessitates a substantial amount of initial setup and fine-tuning to guarantee that it performs appropriately in a variety of situations.

Future Work

The application of sophisticated machine learning models to forecast the possibility of drift has the potential to revolutionize how systems react to configuration inconsistencies in advance. The creation of adaptive systems that operate in real-time: The efficiency of the system might be improved by developing solutions that can react in real-time to drift based on the circumstances and the degree of significance of the changes. Problems with Permanent Changes: As previously stated, some changes, like those changed to storage(for example, once increased cannot be brought back to the lower one) settings, need big steps like deleting stacks or recreating them. It is recommended that future studies concentrate on the development of strategies that enable reversible alterations or recovery methods that are more efficient in these kinds of situations. Remediation Methods That Are Being Improved: To address complex conditions in which typical rollback and synchronization are either insufficient or impossible, research should investigate new remediation strategies that are capable of handling such scenarios.

References

Arifin, M. A., Satra, R., Syafie, L. and Nidhom, A. M. (2023). Optimizing aws lambda code execution time in amazon web services, *Bulletin of Social Informatics Theory and Application* **7**(1): 14–23.

Boscain, S. (2023). *AWS Cloud: Infrastructure, DevOps techniques, State of Art.*, PhD thesis, Politecnico di Torino.

Campbell, B. and Campbell, B. (2020). Cloudformation in-depth, *The Definitive Guide to AWS Infrastructure Automation: Craft Infrastructure-as-Code Solutions* pp. 55–122.

Chijioke-Uche, J. (2022). *Infrastructure as Code Strategies and Benefits in Cloud Computing*, PhD thesis, Walden University.

Cotcharat, S. (2022). *INFRASTRUCTURE AS CODE TOOLS COMPARISON ON AWS CLOUD ENVIRONMENT*, PhD thesis, Dhurakij Pundit University.

Humble, J. and Farley, D. (2010). *Continuous delivery: reliable software releases through build, test, and deployment automation*, Pearson Education.

Kharche, H., Shah, T. and Gautam, T. (2020). Infrastructure as a code-on demand infrastructure, *International Research Journal on Advanced Science Hub* **2**(Special Issue ICARD 2020): 193–197.

Mohanapraneswaran, M., Arun, G. and Prathiksha, M. (n.d.). One scheduler.

Närhi, K. (2023). Aws cdk compared to other iac tools.

Nawagamuwa, J. (2023). Infrastructure as code frameworks evaluation for serverless applications testing in aws.

Pessa, A. (2023). Comparative study of infrastructure as code tools for amazon web services.

Prassanna, J., Pawar, A. and Neelanarayanan, V. (2017). A review of existing cloud automation tools, *Asian J Pharm Clin Res* **10**: 471–473.

Rong, C., Geng, J., Hacker, T. J., Bryhni, H. and Jaatun, M. G. (2022). Openiac: open infrastructure as code-the network is my computer, *Journal of Cloud Computing* **11**(1): 12.

Uzun, B. and Tekinerdogan, B. (2019). Architecture conformance analysis using model-based testing: A case study approach, *Software: Practice and Experience* **49**(3): 423–448.

Uzun, B. and Tekinerdogan, B. (2024). Detecting deviations in the code using architecture view-based drift analysis, *Computer Standards & Interfaces* **87**: 103774.

Valkeinen, M. (2022). *Cloud infrastructure tools for cloud applications: Infrastructure management of multiple cloud platforms*, Master's thesis.

Vasenius, J.-P. (2022). *Migrating manually configured aws infrastructure to use iac approach*, Master's thesis.

Wang, R. (2022). *Infrastructure as Code, Patterns and Practices: With Examples in Python and Terraform*, Simon and Schuster.